CLAIMS:

1. A storage medium containing software for manipulating computer-implemented objects in a distributed system, the software comprising:

code to create a shared environment, the shared environment comprising a plurality of objects; and

code to create an object, the object exposed to other objects in the shared environment, the object comprising:

a set of Behavior logics, each member of the set of Behavior logics adapted to cause the object to perform a task; and

a first Behavior logic, adapted to receive a Command from another object in the shared environment, the first Behavior logic invokable external to the object, the first Behavior logic comprising:

code to receive the Command;

code to select a Behavior logic of the set of Behavior logics corresponding to the Command from a Command-Behavior mapping; and

code to execute the selected Behavior logic responsive to the Command.

2. The storage medium of claim 1, the set of Behavior logics and the Command-Behavior mapping private to the object.

3. The storage medium of claim 1, the set of Behavior logics having no members.

4. The storage medium of claim 1, the object further comprising:

a default Behavior logic, adapted to cause the object to perform a default task, the default Behavior logic private to the object;

the first Behavior logic further comprising:

code to execute the default Behavior logic responsive to the Command if no Behavior logic is selected by the code to select a Behavior logic corresponding to the Command.

5. The storage medium of claim 1, wherein the Command-Behavior mapping can cause the code to select a Behavior to select multiple Behaviors.

6. The storage medium of claim 1, the object further comprising:

an authentication data, the authentication data providable to other objects for authenticating Commands received from the other objects by the code to receive the Command.

7. The storage medium of claim 6, wherein the Command comprises the authentication data, the Command-Behavior mapping restrictable responsive to the authentication data.

8. The storage medium of claim 1, the software further comprising:

code to create a first Shadow of the object, the first Shadow of the object adapted to communicate with the object, the first Shadow of the object being informed of changes to the object and the object being informed of changes to the first Shadow of the object.

9. The storage medium of claim 8, wherein the first Shadow of the object is a copy of the object.

10. The storage medium of claim 8, wherein the Command-Behavior mapping of the first Shadow of the object differs from the Command-Behavior mapping of the object.

11. The storage medium of claim 8, the software further comprising:

code to create a plurality of Shadows of the object adapted to communicate with the object and the first Shadow of the object, the object and the first Shadow of the object being informed of changes to any of the plurality of Shadows of the object and each of the plurality of Shadows of the object being informed of changes to the object and changes to the first Shadow of the object.

12. The storage medium of claim 8, the software further comprising:

code to promote the first Shadow of the object into a new object.

13. The storage medium of claim 12, the software further comprising:

code to create a plurality of Shadows of the object,

wherein executing the code to promote the first Shadow of the object into a new object converts each of the plurality of Shadows of the object into a Shadow of the new object.

14. The storage medium of claim 12, the shared environment further comprising:

a plurality of servers, the object on a first server of the plurality of servers, the first Shadow of the object on a second server of the plurality of servers; and

code to manage the plurality of servers, executing the code to promote the first Shadow of the object to a new object if the first server experiences a predetermined condition.

15. The storage medium of claim 1, the set of Behavior logics further comprising:

code to modify the Command-Behavior mapping to cause the code to select a Behavior logic responsive to the Command to select a different Behavior logic of the set of Behavior logics.

16. The storage medium of claim 1, the shared environment comprising:

a plurality of servers, the object having a location on one of the plurality of servers, the object acting independent of the location.

17. The storage medium of claim 1, the object further comprising:

code to create the Command-Behavior mapping from an external data source.

18. The storage medium of claim 1, the software capable of using any networking protocol.

19. A method of manipulating a computer-implemented object in a distributed system, the method comprising the steps of:

creating a shared environment; the shared environment comprising a plurality of objects; and

creating an object, the object exposed to other objects in the shared environment, the step of creating an object comprising the step of:

coding a set of Behavior logics, each member of the set of Behavior logics causing the object to perform a task;

manipulating the object, comprising the steps of:

receiving a Command from another object of the plurality of objects in the shared environment;

selecting a Behavior logic of the set of Behavior logics corresponding to the Command from a Command-Behavior mapping; and

executing the selected Behavior logic responsive to the Command.

20. The method of claim 19, wherein the set of Behavior logics and the Command-Behavior mapping are private to the object.

21. The method of claim 19, further comprising the step of:

changing the Command-Behavior mapping, causing the step of selecting a Behavior logic to select a different Behavior logic of the set of Behavior logics responsive to the Command.

22. The method of claim 19, the method further comprising the steps of:

coding a default Behavior logic to cause the object to perform a default task, and

executing the default Behavior logic if no Behavior logic is selected by the step of selecting a Behavior logic.

23. The method of claim 19, the set of Behavior logics having no members.

24. The method of claim 19, wherein the Command-Behavior mapping can cause the step of selecting a Behavior logic to select multiple Behaviors.

25. The method of claim 19, further comprising the steps of:

creating an authentication data for the object.

26. The method of claim 25, the Command comprising the authentication data, the method further comprising the step of:

restricting the Command-Behavior mapping responsive to the authentication data.

27. The method of claim 19, further comprising the step of:

creating a first Shadow of the object, the first Shadow of the object adapted to communicate with the object, the first Shadow of the object being informed of changes to the object and the object being informed of changes to the first Shadow of the object.

28. The method of claim 27, the step of creating the first Shadow of the object comprising the step of:

copying the object.

29. The method of claim 27, the step of creating the first Shadow of the object comprising the step of:

modifying the Command-Behavior logic of the first Shadow of the object.

30. The method of claim 27, further comprising the step of:

creating a plurality of Shadows of the object, adapted to communicate with the object and the first Shadow of the object, the object and the first Shadow of the object being informed of changes to any of the plurality of Shadows of the object and each of the plurality of Shadows of the object being informed of changes to the object and changes to the first Shadow of the object.

31. The method of claim 27, further comprising the step of:

promoting the first Shadow of the object into a new object.

32. The method of claim 31, further comprising the step of:

creating a plurality of Shadows of the object,

converting each of the plurality of Shadows of the object into a Shadow of the new object, responsive to the step of promoting the first Shadow of the object.

33. The method of claim 19, the shared environment comprising:

a plurality of servers;

wherein the object has a location on a first server of the plurality of servers, the object acting independent of the location.

34. The method of claim 19, the shared environment capable of using any networking protocol to communicate with another shared environment.

35. The method of claim 19, further comprising the step of:

creating the Command-Behavior mapping from an external data source.

36. A method of designing an application from configurable objects having Behavior logics capable of performing tasks, the method comprising the steps of:

creating a plurality of objects, each object of the plurality of objects adapted to receive and execute Commands, each object exposed to each other object of the plurality of objects, the step of creating the plurality of objects comprising the steps of:

creating a set of Behavior logics for an object, the set of Behavior logics capable of being an empty set;

mapping members of a first set of Commands to members of the set of Behavior logics;

mapping any Command not a member of the first set of Commands to a default Behavior logic; and

configuring a Command-receiver Behavior logic to receive a Command and execute the Behavior logic corresponding to the Command.

37. The method of claim 36, further comprising the steps of:

creating a Shadow of an object of the plurality of objects, the Shadow configured such that sending a Command to the Shadow causes the object to act as if the Command had been sent to the object.

38. The method of claim 37, each of the plurality of objects having a location on one of a plurality of servers, each of the plurality of objects being independent of the location of each other of the plurality of objects.

39. The method of claim 38, a Shadow of each of the plurality of objects automatically created on each of the plurality of servers other than the server on which the object is located.

40. A processor-based system, comprising:

a first processor; and

a first storage device coupled to the first processor containing a software to manipulate computer-implemented objects in a shared environment, the software comprising:

code to create a shared environment, the shared environment comprising a plurality of objects; and

code to create an object of the plurality of objects, the object exposed to other objects in the shared environment, the object comprising:

a set of Behavior logics, each member of the set of Behavior logics adapted to cause the object to perform a task; and

a first Behavior logic, adapted to receive a Command from another object in the shared environment, the first Behavior logic invokable external to the object, the first Behavior logic comprising:

code to receive the Command;

code to select a Behavior logic of the set of Behavior logics corresponding to the Command from a Command-Behavior mapping; and

code to execute the selected Behavior logic responsive to the Command.

41. The processor-based system of claim 40, the object further comprising:

a default Behavior logic, adapted to cause the object to perform a default task, the default Behavior logic private to the object;

the first Behavior logic further comprising:

code to execute the default Behavior logic responsive to the Command if no Behavior logic is selected by the code to select a Behavior logic corresponding to the Command.

42. The processor-based system of claim 40, wherein the Command-Behavior mapping can cause the code to select a Behavior logic to select multiple Behaviors.

43. The processor-based system of claim 40, further comprising:

an input device coupled to the first processor,

wherein a first object of the plurality of objects is coupled to the input device such that manipulation of the input device sends a Command from the first object to a second object of the plurality of objects without identifying the input device, the second object of the plurality of objects acting responsive to the Command independent of the nature of the input device.

44. The processor-based system of claim 40, further comprising:

an output device coupled to the first processor,

wherein a first object of the plurality of objects is coupled to the input device such that a first object is capable of rendering a second object on the output device without identifying the output device to the second object.

45. The processor-based system of claim 40, further comprising:

a second processor;

a network, coupled to the first processor and the second processor;

a second storage device coupled to the second processor, the second storage device containing the software;

the software further comprising:

code to connect the shared environment to the network;

code to create a Shadow on the second processor of the object on the first processor, the Shadow and the object communicating with each other to inform the Shadow of changes to the object and the object of changes to the Shadow.

46. A software architecture for manipulating computer-implemented objects on a plurality of computers, some of the plurality of computers having input devices and some of the plurality of computers having output devices, the software architecture implemented in an extensible object-oriented language, comprising:

a distributed system, comprising:

a plurality of shared environments, each of the plurality of shared environments executing on a different computer of the plurality of computers, each of the plurality of shared environments comprising:

a CommandReceiver class, the CommandReceiver class comprising:

a set of Behavior private methods, each member of the set of Behavior methods adapted to cause instantiations of the CommandReceiver class to perform a task; and

an executeCommand public method, adapted to receive a Command from an object in the shared environment, the executeCommand public method comprising:

code to receive the Command;

code to select a Behavior private method of the set of Behavior private methods selected corresponding to the Command from a Command-Behavior mapping; and code to execute the selected Behavior private method; and

a Kernel subclass of the CommandReceiver class, the Kernel class comprising:

code to instantiate objects of the CommandReceiver class;

code to destroy objects of the CommandReceiver class.

47. The software architecture of claim 46, further comprising:

a Pawn subclass of the CommandReceiver class, the Pawn subclass comprising:

code to register an instantiation of a Pawn with a Kernel object of the Kernel subclass;

code to determine whether an instantion of the Pawn subclass is a real Pawn or a Shadow Pawn of a real Pawn, and

code to send State information about an instantiation of the Pawn subclass,

wherein Commands received by Shadow Pawns are sent to the real Pawn corresponding to the Shadow Pawn.

48. The software architecture of claim 46, further comprising:

a ControlDevice subclass of the CommandReceiver class corresponding to an input device for receiving input from the input device and sending Commands to other CommandReceiver objects.

49. The software architecture of claim 46, further comprising:

a Construct subclass of the CommandReceiver class corresponding to an output device for rendering objects of the CommandReceiver class with graphical attributes.

50. The software architecture of claim 46, further comprising:

a Console subclass of the CommandReceiver class for allowing a user of the distributed system to instantiate, modify, and destroy objects, and for allowing a user to send Commands to CommandReceiver objects.

51. The software architecture of claim 46, further comprising:

a Nengine subclass of the CommandReceiver class for serializing and deserializing CommandReceiver objects, transmitting and receiving the serialized CommandReceiver object across a network to a Nengine in another shared environment of the distributed system.

52. The software architecture of claim 51, further comprising:

a Node subclass of the CommandReceiver class, an instantiation of the Node subclass corresponding to a Pawn object for representing the Pawn object to a Nengine object for communicating State information corresponding to a Pawn to Shadow Pawns of the Pawn and for communicating Commands sent to a Shadow Pawn to the real Pawn corresponding to the Shadow Pawn.